AN ARCHITECTURE TO SUPPORT AUTONOMY, TELEOPERATION, AND SHARED CONTROL

Dr. Ron Lumia, John C. Fiala, Albert J. Wavering

Robot Systems Division
National Bureau of Standards, Gaithersburg, Maryland 20899

Abstract

Described is an approach to the functional architecture of a telerobot so that autonomy, teleoperation and shared control can all be supported. The system is hierarchically organized where task decomposition, world modeling, and sensory processing are explicitly represented. Goals at each level of the hierarchy are decomposed spatially and temporally into simpler tasks which become goals for lower levels. The spatial decomposition facilitates control and coordination of multi-arm robots. A description of the lowest level, the Servo Level, is presented, along with the operator control interface at that level.

1. Introduction

A functional architecture for telerobot control systems is described by Albus, et. al., in [1]. While other recent robot control system descriptions [2,3] have dealt primarily with hardware and operating system structures, the architecture described in [1] emphasizes a modular software structure for all levels of control in an autonomous agent. In addition, the control system design incorporates teleoperation at several levels of control.

The telerobot architecture consists of four control levels, Task, Elemental-Move, Primitive and Servo, arranged hierachically as described in [1]. The Servo Level is at the bottom of the hierarchy and is directly responsible for computing the control outputs to the actuators. The basic structure of the Servo Level for a device, e.g. a manipulator or articulated end-effector, is shown in Figure 1. The device control level is composed of three main components, Sensory Processing (SP), World Modeling (WM), and Task Decomposition (TD). Generally, there is a Servo Level like this for each controlled device of the telerobot [1,5]. To simplify the discussion, the term telerobot will refer in the following only to a manipulator of the telerobot.

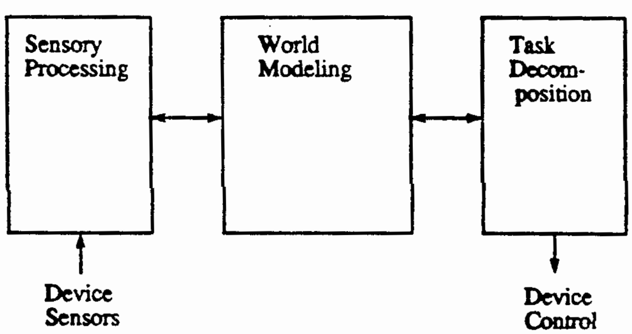

Figure 1. Device Control Architecture.

The Sensory Processing component of the control system is responsible for reading system sensors, and then filtering and integrating this information over space and time. The Task Decomposition component computes the required control outputs. World Modeling obtains new information about the environment from Sensory Processing and provides the latest estimates to Task Decomposition. One of the primary activities of World Modeling is to maintain the *global data system* in which is stored the system's best estimate of the state of the world.

It is desired that a human operator be interfaced to the Servo Level of the telerobot. This interface should allow three principal control modes, *autonomous control*, *teleoperation*, and *shared control*. *Autonomous control* refers to operation without human intervention at the Servo Level. In this mode, the Servo Level of the telerobot receives commands only from the Primitive Level of the control hierarchy. *Teleoperation* refers to control of the telerobot directly by the operator, as in bilateral master-slave control. The third mode, *shared control*, is a combination of the purely autonomous and teleoperated modes. In this mode, the control of the telerobot is determined from a combination of Primitive and operator inputs.

The purpose of this paper is to describe in detail the interfaces and functional requirements for achieving these three control modes of a manipulator at the Servo Level. The control structure for the *teleoperation device*, i.e. joystick or master arm, controller is discussed in Section 2. Section 3 describes some details of the telerobot control structure. The interfaces to the telerobot Servo Level are examined in detail in Sections 4 and 5. Finally, Section 6 describes the functionality of the telerobot Servo Level in achieving the three control modes.

2. Teleoperation Device Architecture

In the general case, a teleoperation device may be actively controlled. This is usually done to provide what has come to be called *force feedback* to the operator. Although force may not be the feedback variable, the idea is that active control of the teleoperation device based on the state of the telerobot can provide the operator with more information on the progress of the telemanipulation task. Many devices, such as simple joysticks, are not actively controlled and provide no feedback to the operator. Devices like JPL's Force-Reflecting Hand Controller [4,6] are specially designed to provide operator feedback. Such devices require active control. Thus, the general architecture for a teleoperation device is the same as that of a control system as described in [1]. This type of architecture is depicted in Figure 1.

With this control structure, Sensory Processing routines read the device sensors and Task Decomposition computes the control outputs to the device. (If the device is not actively controlled, i.e. no force feedback, then the Task Decomposition module is obviously a "null" module.) World Modeling updates the data system as to the current state of the teleoperation device based on data collected by Sensory Processing. Data in the global data system is available to all modules at all levels of the control system. Thus, the data system provides a communication mechanism for widely separated components of the control system that does not interfere with the system's overall modularity. In particular, this mechanism serves to connect the Servo Level of the telerobot with the teleoperation device control level.

Figure 2 depicts the interaction between the telerobot Servo Level and the teleoperation device control system. As shown in the figure, the teleoperation device control system updates the global data system as to the current state of the teleoperation device. This may include the teleoperation device's position, velocity and sensed forces. When in teleoperation mode it is

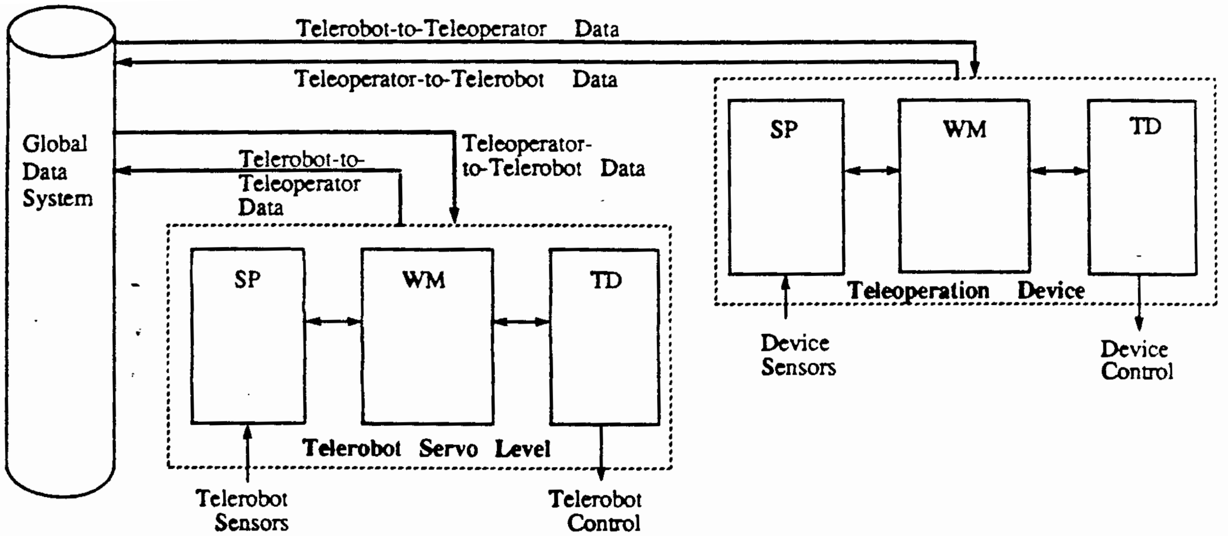

Figure 2. Teleoperation Device/Telerobot Logical Architecture.

desired that the telerobot track the teleoperation device state. Thus, the telerobot World Modeling module will obtain the state of the teleoperation device from the global data system and provide this information to Task Decomposition, where it is used in computing the control outputs for the manipulator. Likewise, the state of the telerobot can be obtained from the data system by the teleoperation device World Modeling module and used to compute the force feedback to the teleoperator.

3. Telerobot Servo Level Architecture

As with the teleoperation device, the telerobot Servo Level has the general structure described in Section 1. However, additional structure can be described within the Sensory Processing, World Modeling, and Task Decomposition components. To limit the discussion, the following will emphasize the element of Task Decomposition involved in allowing both teleoperation and autonomy, the Job Assignment module.

Figure 3 shows the gross structure of the telerobot Servo Level Task Decomposition module. There are three modules. The Execution module directly computes the motor command for the manipulator. To do this, it receives periodic attractor points for the manipulator from the Planning module and model-based terms, (i.e. Jacobians, inertias, etc.,) from World Modeling. This is described in some detail in [5]. The Planning module feeds the periodic data points to the Execution module, performing some simple interpolation on the data it receives from Job Assignment if necessary. The Job Assignment module receives the commands from Primitive and the teleoperator, and provides a coordinated output command to the Planning module.

The Primitive Level of the control hierarchy is responsible for generating the time sequence of Servo Level commands needed to produce a dynamic trajectory. Each command to Servo is basically one point of a position/force trajectory. By sending a sequence of such points Primitive can move the manipulator along a desired path. The Servo Level servos to each goal point using an algorithm specified by Primitive.

The data contained in the Job Assignment module interfaces are shown in the figure. The Primitive/Servo interface contains data passed between the Execution module at Primtive and the Job Assignment module at the Servo Level. The operator control interface contains the teleoperator-to-telerobot data as depicted in Figure 2. Detailed explanations of the data are given below.

4. Primitive/Servo Interface

The parameter C_z is a coordinate system specifier. C_z indicates the coordinate system in which the position and force command vectors are expressed. This is the *servo coordinate system*. The choices for C_z include joint coordinates, (Cartesian) world coordinates, (Cartesian) end-effector coordinates.

The vectors z_d, $\dot{z}_d$, $\ddot{z}_d$, $\dddot{z}_d$, specify the desired position, velocity, acceleration and jerk for the manipulator each control cycle. The vector f_d specifies the desired forces for the manipula-

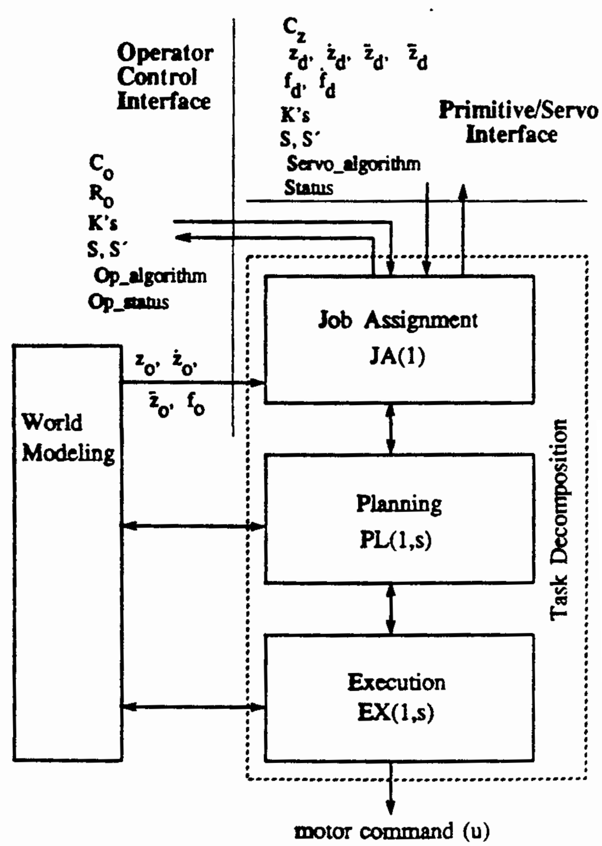

Figure 3. Interfaces to Telerobot Servo Level.

tor. The vector $\dot{f}_d$ specifies the desired force rates. All of these vectors are specified with respect to the coordinate system of C_z. For example, if C_z indicates joint coordinates, then the f_d and $\dot{f}_d$ vectors are vectors of joint torques and torque rates, and the z_d, $\dot{z}_d$, and $\ddot{z}_d$, are joint positions, velocities, accelerations. The set of six vectors (z_d, $\dot{z}_d$, $\ddot{z}_d$, $\dddot{z}_d$, f_d, $\dot{f}_d$) forms the attractor set, (desired state,) of the manipulator for each control cycle.

The gain terms (K's) are the gain coefficients which multiply the error vectors in the control equations. The K's determine the impedance of the manipulator [5]. The parameters S and S′ are selection matrices used to select the coordinate axes to be force controlled and position controlled [5].

The final element of the Servo command input is the "Servo_algorithm" selector. This parameter indicates what algorithm is to be used for approaching the attractor set during subsequent control cycles. The "Status" parameter indicates the status of the Servo Level control and is passed back to Primitive from the Job Assignment module.

5. Operator Control Interface

The coordinate system specification from the operator, C_o, indicates the coordinates in which the operator commands are to be interpreted. The possible values of C_o take the same form as those of C_z described for the Primitive/Servo interface.

The actual command positions, velocities, accelerations, and forces, z_o, $\dot{z}_o$, $\ddot{z}_o$, f_o, from the operator are generated from the inputs of the teleoperation device. The possible devices from which operator input could be taken include joysticks, control pendants of various forms, and master arms.

The redundancy resolution parameter R_o specifies how redundancy is to be resolved for the operator command. This is probably only required when doing shared control, i.e. the operator coordinates C_o are underspecified with respect to the autonomous coordinates C_z. Normally, the coordinates of the command input to Servo are the coordinates of the servo control algorithm. In this case, any redundancy resolution needed to transform the command into joint space is incorporated in the dynamics of the control algorithm. The Servo Level does not make a transformation of coordinates from command to servo coordinates, because the command is in the desired servo coordinates. However, during shared control, a transformation may be required to bring the Primitive input and the operator input into the same coordinate system. This may require redundancy resolution independent of the servo algorithm.

The servo gains (K's) and selection matrices (S,S′) are input from the operator interface when the command to Servo is taken from the operator. These parameters are of exactly the same form as the corresponding parameters in the Primitive to Servo interface. During shared control, the Job Assignment module must chose one set of these parameters based on the Op_algorithm.

The overall control of the Servo Level is achieved by the parameter Op_algorithm. The Op_algorithm parameter selects between the autonomous, teleoperation, and shared control modes. The parameter Op_status is analogous to the Status returned to Primitive by the Job Assignment module. The Op_status informs the operator control of the current status of the Servo Level.

The Job Assignment module receives from World Modeling the command data from the operator's input device. These vectors give the desired position, velocity, acceleration, and force for the manipulator when the Servo Level is in teleoperation mode. Note that these vectors are with respect to the coordinates selected by C_o for pure teleoperation, and with respect to the coordinates selected by C_z for shared control. Thus, World Modeling may have to make the transformation from C_o to C_z.

To clarify the meaning of this general interface it is helpful to examine a specific example. For the purpose of discussion consider teleoperation devices as divided into two classes. The first class consists of devices which are kinematically similar to the telerobot or otherwise provide state information in the joint-space of the telerobot. These are the *joint-space teleoperation devices*. They include identical master-slave arm configurations and the so-called mini-master devices. Joysticks can operate in joint coordinates as well, albeit less conveniently. The second class of devices consists of all devices providing data in coordinate systems linked to some Cartesian frame of reference. These devices are the *Cartesian teleoperation devices*, which include JPL's Force-Reflecting Hand Controller and Cartesian joysticks such as DFVLR's sensor ball device [7]. Tables 1 and 2 give the information in the data paths of Figure 2 as required for Cartesian teleoperation devices performing pure teleoperation.

Table 1 shows the information in the teleoperator-to-telerobot data path for Cartesian teleoperation. The inputs z_o, $\dot{z}_o$ and f_o are all derived from the movement of the joystick or hand-controller part of the teleoperation input. The remaining parameters of the interface are obtained from the operator either through switches or a terminal.

The number of data items that need to be passed for each element of the interface is also given in the table. Each item is a 32-bit data word, usually in floating point format. For example, the Cartesian velocity vectors in the interface have the form

$$[\, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z \,],$$

where v_x, v_y, and v_z are the linear velocity components of the end-effector motion with respect to the x, y, and z axes of the control coordinates, and ω_x, ω_y, and ω_z are the angular velocity components about the same axes. Thus, velocity requires six data words. The six-dimensional Cartesian force vectors are defined analogously.

The definition of the position vectors is not as straight-forward. To avoid the ambiguities of using only three orientation parameters, the orientation part of the position vector is represented by an equivalent angle-axis form [8]. Thus, the form of the position vector is

$$[\, x, y, z, \theta, n_x, n_y, n_z],$$

where x, y, and z give the position with respect to the origin of the control coordinates, and the orientation is given by a rotation θ about the unit vector n in the same coordinate system.

The parameter C_o specifies the coordinate system in which the control is to be executed. The general form of C_o is { coord. sys., T_w, T_e }. Here, "coord. sys." is one data word and indicates the selection of "world" or "end-effector" coordinates. *World coordinates* mean that the command vectors give the position of the end-effector with respect to a coordinate system fixed at the manipulator base. *End-effector coordinates* mean that the

Table 1. Cartesian Teleoperator-to-Telerobot Data.

Data	Nature of Data	# items
z_o	End effector positions from device	7
$\dot{z}_o$	End effector velocities from device	6
f_o	End effector forces from device	6
Op_algorithm	Control mode from operator	1
C_o	Coordinate system specifier	25
$K_p, K_v, K_i,$ K_{pf}, K_{vf}, K_{if}	Control gains for telerobot	36
S, S'	Control selection matrices	36

Table 2. Cartesian Telerobot-to-Teleoperator Data.

Data	Nature of Data	# items
z	Telerobot end effector positions	7
$\dot{z}$	Telerobot end effector velocities	6
f	Telerobot end effector forces	6
Status	Status of telerobot control	1
θ	Telerobot joint positions	# joints

operator sees the teleoperation device movements interpreted with respect to a coordinate system fixed at the end-effector. Obviously, the Cartesian position vectors are not used when in end-effector coordinates. The transformations T_w and T_e allow the operator to position the coordinate frames fixed at the base and at the end-effector arbitrarily, as depicted in Figure 4. Each transformation is a rotation matrix and a translation vector, requiring twelve data words. Thus, the C_o parameter lets the operator move in any desired Cartesian system, (and even Cylindrical or other coordinates are possible.) Any scaling or indexing of the teleoperation commands performed for the convenience of the operator should be handled in the World Modeling module of the teleoperation device. Thus, the teleoperation commands are ready to be used in the control when they enter the global data system.

The control gain matrices are six-by-six diagonal matrices and thus require six data items per matrix. The use of these six gain terms is described in [5]. The selection matrices take the most general form as task specification matrices in [9]. Each matrix may have up to eighteen non-zero elements.

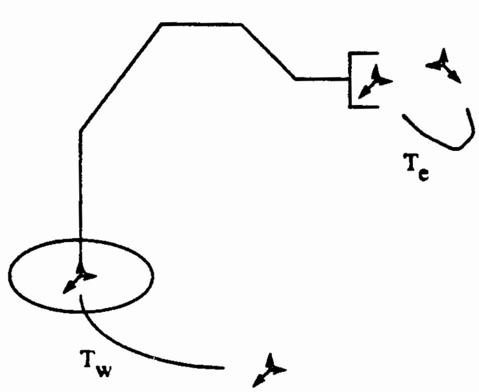

Figure 4. Cartesian Coordinate System Relationships.

The information returned to the teleoperation device controller along the telerobt-to-teleoperator data path is given in Table 2. The telerobot controller returns a Status word and the state vectors z, $\dot{z}$, and f. These Cartesian vectors give the current state of the telerobot in coordinates C_o. The vector f represents World Modeling's "best guess" of the forces in the these coordinates. This information may represent a fusion from a number of sensors including wrist force/torque sensors, joint torque sensors, and tactile force sensors.

One important consideration regarding Cartesian teleoperation is the handling of singularities. When the servo algorithm uses Cartesian inputs, motion along a nearly singular direction can produce a large joint space output. During autonomous operation it is the Primitive Level's job to insure that input commands do not exercise a singularity. This same criterion applies to the inputs from a Cartesian teleoperation device – the inputs to Servo should not ask for motions along singular directions.

Note that this does not mean that the teleoperator can not move the telerobot into a singular region. However, in a singular region, the operator's command input should be restricted to movement which does not exercise the singularity. This means that the Cartesian input is scaled according to the manipulability measure, and, at the point of singularity, reduced to the remaining degrees of freedom. For a Cartesian teleoperation device with force feedback this means that the manipulability ellipsoid [10] should be reflected back to the operator such that the operator *senses* the mechanism's singular regions. In order for the teleoperation device to specify the correct inputs it must have a local model of the telerobot. The correct inputs can be determined by using the fedback joint positions θ of the telerobot and the local telerobot model.

The data rates required to support teleoperation using the example interface is large. The smallest reasonable subset of this interface would be $\{z_o$, Op_algorithm, z, Status$\}$. If this data was to be exchanged every 20 ms., a serial rate of at least 25.6 Kbaud would be required. For the general teleoperation case, 114 32-bit words passed in 10 ms., a serial data rate of approximately 0.5 Mbaud would be required. Note that all values need not be passed every cycle. For instance, the K's, S's, Op_algorithm, and C_o parameters will only change occasionally, and thus can be passed less frequently than the other data. For this reason, these parameters can appear as optional parameters at the end of the communication buffer. They would only be transferred when the values change. Still, 32-bit parallel data transfers may be needed for the teleoperation interface.

6. Autonomous, Teleoperated, and Shared Control Modes

It is the task of the Job Assignment module to determine whether Servo will execute from the autonomous commands (Primitive), or from operator commands given through some input device such as a joystick or master arm. In addition, the Job Assignment module must be able to coordinate operator and Primitive commands for shared control operation.

The Job Assignment module, upon receiving a new command, first examines the Op_algorithm. This parameter indicates the desired control mode selected by the operator. The control mode could be autonomous, teleoperated, or shared control.

For autonomous control mode, the Job Assignment module takes the input strictly from the Primitive/Servo interface. No control data is obtained from the operator. Thus, as stated previously, the control algorithm is given by Servo_algorithm. Also, in the autonomous mode, no data is obtained from world modeling to modify the Primitive command vectors. Operation in this mode is described in [5].

The Job Assignment module takes the Op_algorithm as the control algorithm in the teleoperated control mode. In this mode, control data comes from the operator control interface as depicted in Figure 3. For Cartesian teleoperation, using the data of Tables 1 and 2, a control of the form

$$u_{telerobot} = J^t(\theta)\ [\ K_p(z_o - z) + K_v(\ \dot{z}_o - \dot{z})\ ]$$

could be computed in the telerobot Servo Level Execution module. (The $J^t(\theta)$ matrix is the manipulator Jacobian; see [8].) At the same time a force feedback control could be computed for the teleoperation device by the teleoperation device Execution module. The technique given in [4] for the Force-Reflecting Hand Controller is

$$u_{device} = K_{fm}\ J^t(\theta_m)\{\ K_{pm}\mathrm{Dbnd}(z - z_m) - K_{vm}\dot{z}_m + f\ \},$$

where z_m is the position of teleoperation device in C_o, and the function Dbnd( ) is a deadband function which eliminates corrections for position errors of small magnitude. The six-dimensional error term $(z - z_m)$ can be computed from the seven-dimensional position form [9].

A number of different controls are possible for the teleoperation mode as described in [5], including unilateral master-slave control, unilateral joystick control, and generalized bilateral control [4,6].

The Op_algorithm could also indicate that some kind of shared control is to be executed. Although the term "shared control" has begun to appear frequently in the literature, it is defined here in terms of the structure of this control system. Shared control occurs at the Servo Level when both the Primitive command and the operator contribute to the final control output from the Job Assignment module. For example, suppose the operator desires that the telerobot's position be fixed in the x- and y-directions of a world coordinate system, while the operator moves the manipulator along the z-axis. This would not required "shared" control since the teleoperation device controller need only fix the x and y values of $\dot{z}_o$ of the teleoperator-to-tele-robot input to achieve this. Although the telerobot Servo Level is handling part of the control, it is not doing any more than is required for generalized bilateral control anyway. This differs somewhat from the description of shared control, termed "interactive manual-automatic control", in [6].

An example of shared control is when the manipulator is commanded by Primitive to track an object or follow a predefined trajectory and the operator is allowed to modify the trajectory at the Servo Level. Such a system is described in [7]. In this mode, the control algorithm would be determined from both Op_algorithm and Servo_algorithm. The servo gains and selection matrices will be chosen from one of the two interfaces. Also, the final attractor set for the manipulator would be a combination of the attractors specified by World Modeling (from the operator) and Primitive. In this mode, the operator command vectors can represent modifications to be made to the autonomous command vectors coming from Primitive. For example, in a rate control mode, the final rate $\dot{z}_s$ for the manipulator could be computed as

$$\dot{z}_s = \dot{z}_d + \dot{z}_o,$$

by the Job Assignment module. Here, $\dot{z}_o$ is a modification of the Primitive rate command by the operator.

Another type of shared control be achieved by choosing elements from both Primitive and the operator to create a new attractor set. For instance, in a hybrid position/force control scheme, the commands computed by Primitive could be used for the force-controlled subspace, and the position-controlled subspace could be given by the operator's input.

7. Conclusion

The architecture presented in [1] provides a complete, modular structure for an advanced telerobot control system. This same structure is applicable to other major systems to be used with the telerobot. One example is the control/sensor system of the teleoperation device(s), as described in this paper. Given these generic control structures it is possible to define interfaces that allow autonomous, teleoperated, and shared control modes all in the same system. This document has presented, as an example, a basic teleoperation interface for use with Cartesian teleoperation devices. Such interfaces support numerous algorithms for telerobot control.

8. References

[1] Albus, J. S., McCain, H. G., Lumia, R., NASA/NBS Standard Reference Model Telerobot Control System Architecture (NASREM), NBS Technical Note 1235, July, 1987.

[2] Bejczy, A. K., Szakaly, Z., "Universal Computer Control System (UCCS) for Space Telerobots," IEEE Conf. Robotics & Automation, Raleigh, N. C., March, 1987.

[3] Kazanzides, P., Wasti, H., Wolovich, W. A., "A Multiprocessor System for Real-time Robotic Control: Design and Applications," IEEE Conf. Robotics & Automation, Raleigh, N. C., March, 1987.

[4] Handlykken, M., Turner, T., "Control System Analysis and Synthesis for a Six Degree-of-Freedom Universal Force-Reflecting Hand Controller," Nineteenth IEEE Conf. Decision & Control, Albuquerque, N.M., Dec., 1980.

[5] Fiala, J. C., Lumia, R., Albus, J. S., "Servo Level Algorithms for the NASREM Telerobot Control System Architecture," Proc. of SPIE Vol.851 - Space Station Automation III, Cambridge, Mass., Nov., 1987.

[6] Bejczy, A. K., "Robots as Man-Extension Systems in Space," IFAC Ninth Triennial World Congress, Budapest, Hungary, 1984.

[7] Hirzinger, G., "Robot Learning and Teach-In Based on Sensory Feedback," Third Symp. Robotics Research, Gouvieux, France, 1985.

[8] Craig, J. J., Introduction to Robotics; Mechanics and Control, Addison-Wesley, Reading, Mass., 1986, p. 46.

[9] Khatib, O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE Jour. Robotics & Automation, Vol. RA-3, No. 1, Feb., 1987.

[10] Yoshikawa, T., "Manipulability and Redundancy Control of Robotic Mechanisms," IEEE Conf. Robotics & Automation, St. Louis, March, 1985.